RFC: 760
IEN: 128

DOD STANDARD

INTERNET PROTOCOL

January 1980

prepared for

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

by

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

TABLE OF CONTENTS

PREFACE iii

1. INTRODUCTION 1

1.1 Motivation 1
1.2 Scope 1
1.3 Interfaces 1
1.4 Operation 2

2. OVERVIEW 5

2.1 Relation to Other Protocols 5
2.2 Model of Operation 5
2.3 Function Description 7

3. SPECIFICATION 11

3.1 Internet Header Format 11
3.2 Discussion 21
3.3 Examples & Scenarios 30
3.4 Interfaces 34

GLOSSARY 37

REFERENCES 41

PREFACE

This document specifies the DoD Standard Internet Protocol. This document is based on five earlier editions of the ARPA Internet Protocol Specification, and the present text draws heavily from them. There have been many contributors to this work both in terms of concepts and in terms of text. This edition revises the details security, compartmentation, and precedence features of the internet protocol.

Jon Postel

Editor

January 1980
RFC: 760
IEN: 128
Replaces: IENs 123, 111,
80, 54, 44, 41, 28, 26

DOD STANDARD

INTERNET PROTOCOL

1. INTRODUCTION

1.1. Motivation

The Internet Protocol is designed for use in interconnected systems of packet-switched computer communication networks. Such a system has been called a "catenet" [1]. The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The internet protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks.

1.2. Scope

The internet protocol is specifically limited in scope to provide the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. There are no mechanisms to promote data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols.

1.3. Interfaces

This protocol is called on by host-to-host protocols in an internet environment. This protocol calls on local network protocols to carry the internet datagram to the next gateway or destination host.

For example, a TCP module would call on the internet module to take a TCP segment (including the TCP header and user data) as the data portion of an internet datagram. The TCP module would provide the addresses and other parameters in the internet header to the internet module as arguments of the call. The internet module would then create an internet datagram and call on the local network interface to transmit the internet datagram.

In the ARPANET case, for example, the internet module would call on a local net module which would add the 1822 leader [2] to the internet datagram creating an ARPANET message to transmit to the IMP. The ARPANET address would be derived from the internet address by the local network interface and would be the address of some host in the ARPANET, that host might be a gateway to other networks.

1.4. Operation

The internet protocol implements two basic functions: addressing and fragmentation.

The internet modules use the addresses carried in the internet header to transmit internet datagrams toward their destinations. The selection of a path for transmission is called routing.

The internet modules use fields in the internet header to fragment and reassemble internet datagrams when necessary for transmission through "small packet" networks.

The model of operation is that an internet module resides in each host engaged in internet communication and in each gateway that interconnects networks. These modules share common rules for interpreting address fields and for fragmenting and assembling internet datagrams. In addition, these modules (especially in gateways) may have procedures for making routing decisions and other functions.

The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise).

The internet protocol uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

The Type of Service is used to indicate the quality of the service desired; this may be thought of as selecting among Interactive, Bulk, or Real Time, for example. The type of service is an abstract or generalized set of parameters which characterize the service choices provided in the networks that make up the internet. This type of service indication is to be used by gateways to select the actual transmission parameters for a particular network, the network to be used for the next hop, or the next gateway when routing an internet datagram.

The Time to Live is an indication of the lifetime of an internet datagram. It is set by the sender of the datagram and reduced at the points along the route where it is processed. If the time to live reaches zero before the internet datagram reaches its destination, the internet datagram is destroyed. The time to live can be thought of as a self destruct time limit.

The Options provide for control functions needed or useful in some situations but unnecessary for the most common communications. The

options include provisions for timestamps, error reports, and special routing.

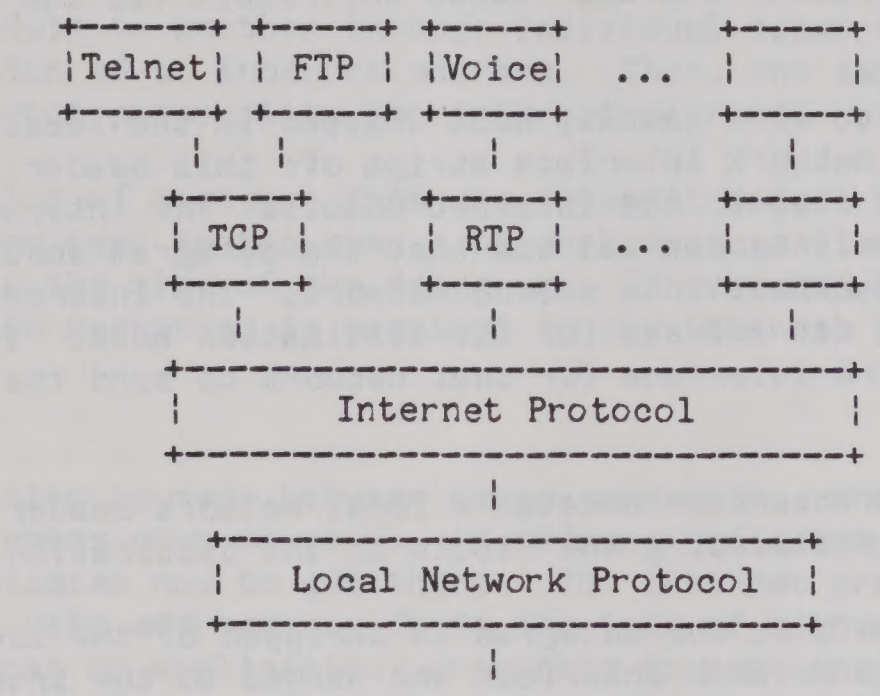
The Header Checksum provides a verification that the information used in processing internet datagram has been transmitted correctly. The data may contain errors. If the header checksum fails, the internet datagram is discarded at once by the entity which detects the error.

The internet protocol does not provide a reliable communication facility. There are no acknowledgments either end-to-end or hop-by-hop. There is no error control for data, only a header checksum. There are no retransmissions. There is no flow control.

2. OVERVIEW

2.1. Relation to Other Protocols

The following diagram illustrates the place of the internet protocol in the protocol hierarchy:



Protocol Relationships

Figure 1.

Internet protocol interfaces on one side to the higher level host-to-host protocols and on the other side to the local network protocol.

2.2. Model of Operation

The model of operation for transmitting a datagram from one application program to another is illustrated by the following scenario:

We suppose that this transmission will involve one intermediate gateway.

The sending application program prepares its data and calls on its local internet module to send that data as a datagram and passes the destination address and other parameters as arguments of the call.

The internet module prepares a datagram header and attaches the data

to it. The internet module determines a local network address for this internet address, in this case it is the address of a gateway. It sends this datagram and the local network address to the local network interface.

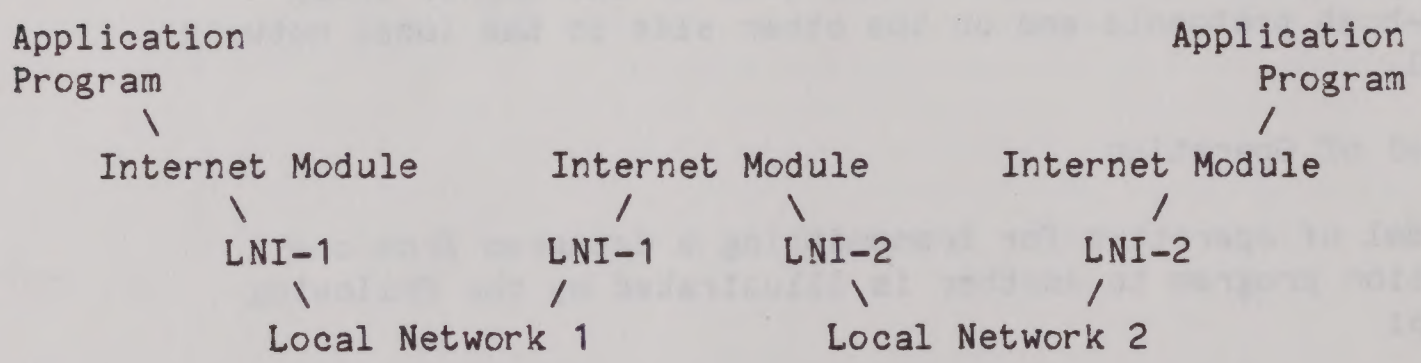
The local network interface creates a local network header, and attaches the datagram to it, then sends the result via the local network.

The datagram arrives at a gateway host wrapped in the local network header, the local network interface strips off this header, and turns the datagram over to the internet module. The internet module determines from the internet address that the datagram should be forwarded to another host in a second network. The internet module determines a local net address for the destination host. It calls on the local network interface for that network to send the datagram.

This local network interface creates a local network header and attaches the datagram sending the result to the destination host.

At this destination host the datagram is stripped of the local net header by the local network interface and handed to the internet module.

The internet module determines that the datagram is for an application program in this host. It passes the data to the application program in response to a system call, passing the source address and other parameters as results of the call.



Transmission Path

Figure 2

2.3. Function Description

The function or purpose of Internet Protocol is to move datagrams through an interconnected set of networks. This is done by passing the datagrams from one internet module to another until the destination is reached. The internet modules reside in hosts and gateways in the internet system. The datagrams are routed from one internet module to another through individual networks based on the interpretation of an internet address. Thus, one important mechanism of the internet protocol is the internet address.

In the routing of messages from one internet module to another, datagrams may need to traverse a network whose maximum packet size is smaller than the size of the datagram. To overcome this difficulty, a fragmentation mechanism is provided in the internet protocol.

Addressing

A distinction is made between names, addresses, and routes [3]. A name indicates what we seek. An address indicates where it is. A route indicates how to get there. The internet protocol deals primarily with addresses. It is the task of higher level (i.e., host-to-host or application) protocols to make the mapping from names to addresses. The internet module maps internet addresses to local net addresses. It is the task of lower level (i.e., local net or gateways) procedures to make the mapping from local net addresses to routes.

Addresses are fixed length of four octets (32 bits). An address begins with a one octet network number, followed by a three octet local address. This three octet field is called the "rest" field.

Care must be taken in mapping internet addresses to local net addresses; a single physical host must be able to act as if it were several distinct hosts to the extent of using several distinct internet addresses. A host should also be able to have several physical interfaces (multi-homing).

That is, a host should be allowed several physical interfaces to the network with each having several logical internet addresses.

Examples of address mappings may be found in reference [4].

Fragmentation

Fragmentation of an internet datagram may be necessary when it originates in a local net that allows a large packet size and must

traverse a local net that limits packets to a smaller size to reach its destination.

An internet datagram can be marked "don't fragment." Any internet datagram so marked is not to be internet fragmented under any circumstances. If internet datagram marked don't fragment cannot be delivered to its destination without fragmenting it, it is to be discarded instead.

Fragmentation, transmission and reassembly across a local network which is invisible to the internet protocol module is called intranet fragmentation and may be used [5].

The internet fragmentation and reassembly procedure needs to be able to break a datagram into an almost arbitrary number of pieces that can be later reassembled. The receiver of the fragments uses the identification field to ensure that fragments of different datagrams are not mixed. The fragment offset field tells the receiver the position of a fragment in the original datagram. The fragment offset and length determine the portion of the original datagram covered by this fragment. The more-fragments flag indicates (by being reset) the last fragment. These fields provide sufficient information to reassemble datagrams.

The identification field is used to distinguish the fragments of one datagram from those of another. The originating protocol module of an internet datagram sets the identification field to a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the internet system. The originating protocol module of a complete datagram sets the more-fragments flag to zero and the fragment offset to zero.

To fragment a long internet datagram, an internet protocol module (for example, in a gateway), creates two new internet datagrams and copies the contents of the internet header fields from the long datagram into both new internet headers. The data of the long datagram is divided into two portions on a 8 octet (64 bit) boundary (the second portion might not be an integral multiple of 8 octets, but the first must be). Call the number of 8 octet blocks in the first portion NFB (for Number of Fragment Blocks). The first portion of the data is placed in the first new internet datagram, and the total length field is set to the length of the first datagram. The more-fragments flag is set to one. The second portion of the data is placed in the second new internet datagram, and the total length field is set to the length of the second datagram. The more-fragments flag carries the same value as the long datagram. The fragment offset field of the second new internet

datagram is set to the value of that field in the long datagram plus NFB.

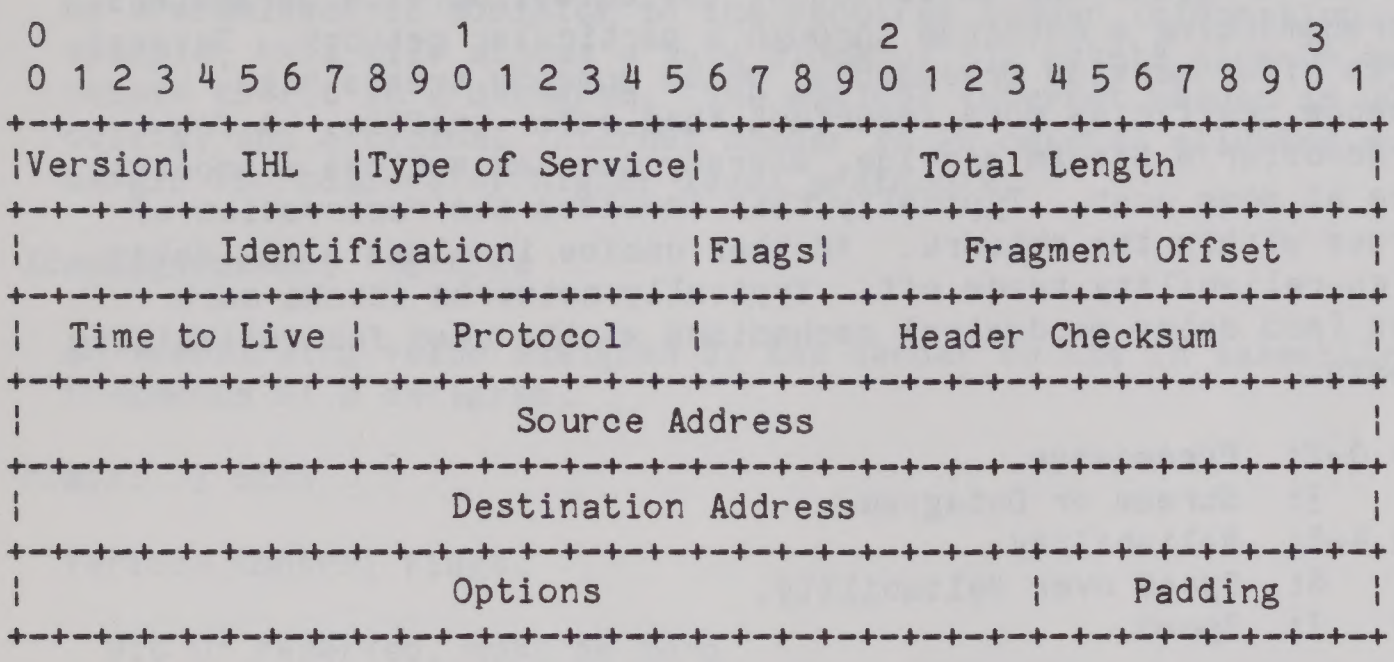
This procedure can be generalized for an n-way split, rather than the two-way split described.

To assemble the fragments of an internet datagram, an internet protocol module (for example at a destination host) combines internet datagram that all have the same value for the four fields: identification, source, destination, and protocol. The combination is done by placing the data portion of each fragment in the relative position indicated by the fragment offset in that fragment's internet header. The first fragment will have the fragment offset zero, and the last fragment will have the more-fragments flag reset to zero.

3. SPECIFICATION

3.1. Internet Header Format

A summary of the contents of the internet header follows:



Example Internet Datagram Header

Figure 3.

Note that each tick mark represents one bit position.

Version: 4 bits

The Version field indicates the format of the internet header. This document describes version 4.

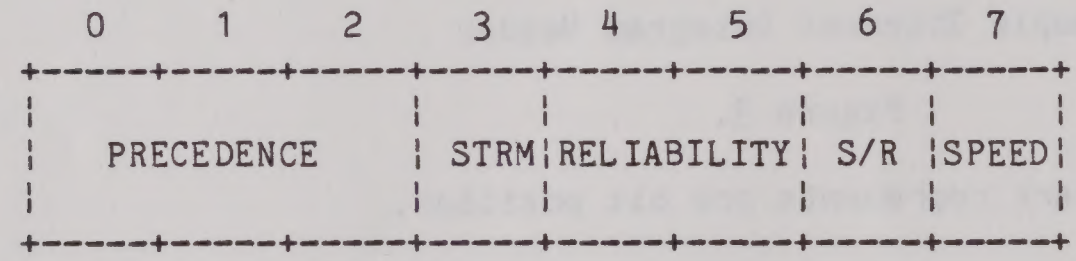
IHL: 4 bits

Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.

Type of Service: 8 bits

The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network. Several networks offer service precedence, which somehow treats high precedence traffic as more important than other traffic. A few networks offer a Stream service, whereby one can achieve a smoother service at some cost. Typically this involves the reservation of resources within the network. Another choice involves a low-delay vs. high-reliability trade off. Typically networks invoke more complex (and delay producing) mechanisms as the need for reliability increases.

Bits 0-2: Precedence.
Bit 3: Stream or Datagram.
Bits 4-5: Reliability.
Bit 6: Speed over Reliability.
Bits 7: Speed.



PRECEDENCE	STRM	RELIABILITY	S/R	SPEED
111-Flash Override	1-STREAM	11-highest	1-speed	1-high
110-Flash	0-DTGRM	10-higher	0-rlblt	0-low
11X-Immediate		01-lower		
01X-Priority		00-lowest		
00X-Routine				

The type of service is used to specify the treatment of the datagram during its transmission through the internet system. In the discussion (section 3.2) below, a chart shows the relationship of the internet type of service to the actual service provided on the ARPANET, the SATNET, and the PRNET.

Total Length: 16 bits

Total Length is the length of the datagram, measured in octets, including internet header and data. This field allows the length of a datagram to be up to 65,535 octets. Such long datagrams are impractical for most hosts and networks. All hosts must be prepared to accept datagrams of up to 576 octets (whether they arrive whole

or in fragments). It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams.

The number 576 is selected to allow a reasonable sized data block to be transmitted in addition to the required header information. For example, this size allows a data block of 512 octets plus 64 header octets to fit in a datagram. The maximal internet header is 60 octets, and a typical internet header is 20 octets, allowing a margin for headers of higher level protocols.

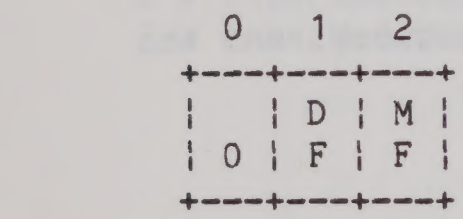
Identification: 16 bits

An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Flags: 3 bits

Various Control Flags.

Bit 0: reserved, must be zero
Bit 1: Don't Fragment This Datagram (DF).
Bit 2: More Fragments Flag (MF).



Fragment Offset: 13 bits

This field indicates where in the datagram this fragment belongs. The fragment offset is measured in units of 8 octets (64 bits). The first fragment has offset zero.

Time to Live: 8 bits

This field indicates the maximum time the datagram is allowed to remain the internet system. If this field contains the value zero, then the datagram should be destroyed. This field is modified in internet header processing. The time is measured in units of seconds. The intention is to cause undeliverable datagrams to be discarded.

Protocol: 8 bits

This field indicates the next level protocol used in the data portion of the internet datagram. The values for various protocols are specified in reference [6].

Header Checksum: 16 bits

A checksum on the header only. Since some header fields may change (e.g., time to live), this is recomputed and verified at each point that the internet header is processed.

The checksum algorithm is:

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

This is a simple to compute checksum and experimental evidence indicates it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Source Address: 32 bits

The source address. The first octet is the Source Network, and the following three octets are the Source Local Address.

Destination Address: 32 bits

The destination address. The first octet is the Destination Network, and the following three octets are the Destination Local Address.

Options: variable

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option:

Case 1: A single octet of option-type.

Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

The option-length octet counts the option-type octet and the option-length octet as well as the option-data octets.

The option-type octet is viewed as having 3 fields:

1 bit reserved, must be zero
2 bits option class,
5 bits option number.

The option classes are:

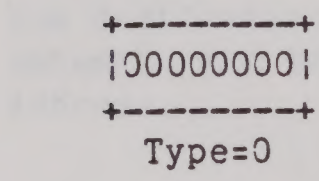
0 = control
1 = internet error
2 = experimental debugging and measurement
3 = reserved for future use

The following internet options are defined:

CLASS	NUMBER	LENGTH	DESCRIPTION
0	0	-	End of Option list. This option occupies only 1 octet; it has no length octet.
0	1	-	No Operation. This option occupies only 1 octet; it has no length octet.
0	2	4	Security. Used to carry Security, and user group (TCC) information compatible with DOD requirements.
0	3	var.	Source Routing. Used to route the internet datagram based on information supplied by the source.
0	7	var.	Return Route. Used to record the route an internet datagram takes.
0	8	4	Stream ID. Used to carry the stream identifier.
1	1	var.	General Error Report. Used to report errors in internet datagram processing.
2	4	6	Internet Timestamp.
2	5	6	Satellite Timestamp.

Specific Option Definitions

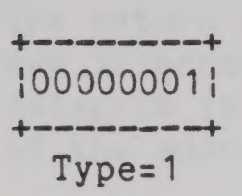
End of Option List



This option indicates the end of the option list. This might not coincide with the end of the internet header according to the internet header length. This is used at the end of all options, not the end of each option, and need only be used if the end of the options would not otherwise coincide with the end of the internet header.

May be copied, introduced, or deleted on fragmentation.

No Operation

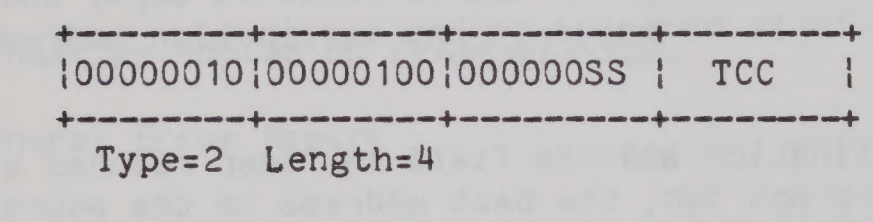


This option may be used between options, for example, to align the beginning of a subsequent option on a 32 bit boundary.

May be copied, introduced, or deleted on fragmentation.

Security

This option provides a way for DOD hosts to send security and TCC (closed user groups) parameters through networks whose transport leader does not contain fields for this information. The format for this option is as follows:



Security: 2 bits

Specifies one of 4 levels of security

11-top secret
10-secret
01-confidential
00-unclassified

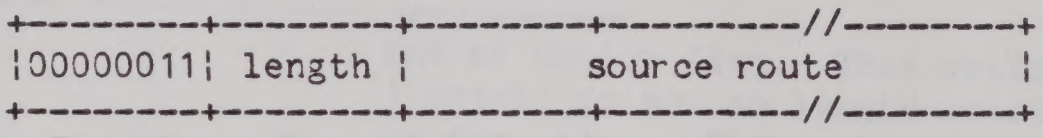
Transmission Control Code: 8 bits

Provides a means to compartmentalize traffic and define controlled communities of interest among subscribers.

Note that this option does not require processing by the internet module but does require that this information be passed to higher level protocol modules. The security and TCC information might be used to supply class level and compartment information for transmitting datagrams into or through AUTODIN II.

Must be copied on fragmentation.

Source Route



Type=3

The source route option provides a means for the source of an internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination.

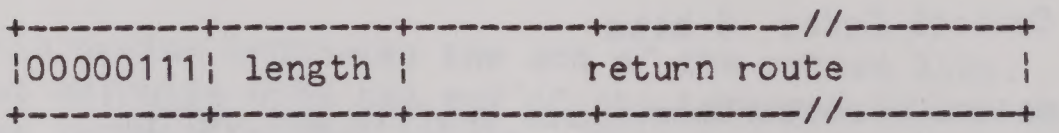
The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, as well as length-2 octets of source route data.

A source route is composed of a series of internet addresses. Each internet address is 32 bits or 4 octets. The length defaults to two, which indicates the source route is empty and the remaining routing is to be based on the destination address field.

If the address in destination address field has been reached and this option's length is not two, the next address in the source route replaces the address in the destination address field, and is deleted from the source route and this option's length is reduced by four. (The Internet Header Length Field must be changed also.)

Must be copied on fragmentation.

Return Route



Type=7

The return route option provides a means to record the route of an internet datagram.

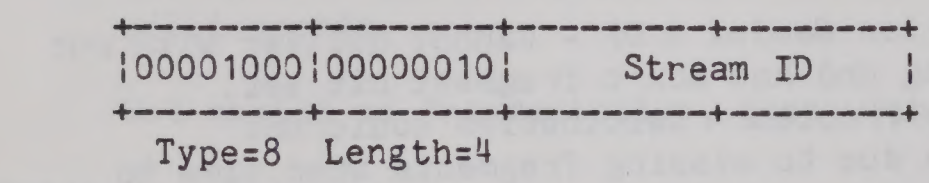
The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, as well as length-2 octets of return route data.

A return route is composed of a series of internet addresses. The length defaults to two, which indicates the return route is empty.

When an internet module routes a datagram it checks to see if the return route option is present. If it is, it inserts its own internet address as known in the environment into which this datagram is being forwarded into the return route at the front of the address string and increments the length by four.

Not copied on fragmentation, goes in first fragment only.

Stream Identifier

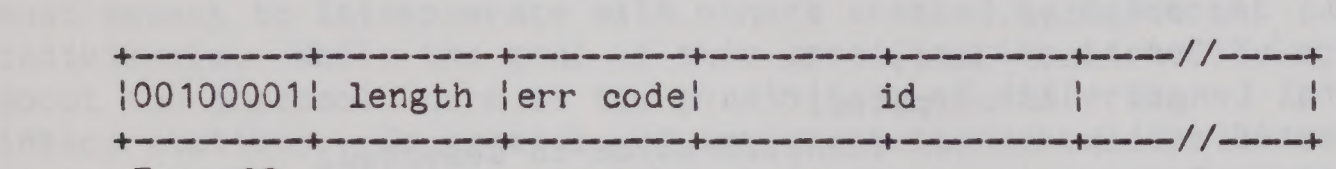


Type=8 Length=4

This option provides a way for the 16-bit SATNET stream identifier to be carried through networks that do not support the stream concept.

Must be copied on fragmentation.

General Error Report



Type=33

The general error report is used to report an error detected in processing an internet datagram to the source internet module of that datagram. The "err code" indicates the type of error detected, and the "id" is copied from the identification field of the datagram in error, additional octets of error information may be present depending on the err code.

If an internet datagram containing the general error report option is found to be in error or must be discarded, no error report is sent.

ERR CODE:

0 - Undetermined Error, used when no information is available about the type of error or the error does not fit any defined class. Following the id should be as much of the datagram (starting with the internet header) as fits in the option space.

1 - Datagram Discarded, used when specific information is

available about the reason for discarding the datagram can be reported. Following the id should be the original (4-octets) destination address, and the (1-octet) reason.

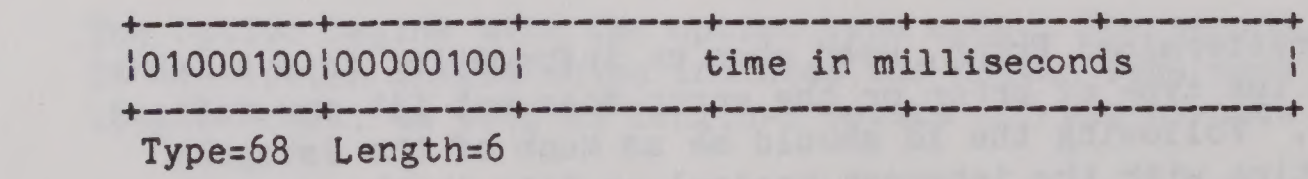
Reason	Description
0	No Reason
1	No One Wants It - No higher level protocol or application program at destination wants this datagram.
2	Fragmentation Needed & DF - Cannot deliver with out fragmenting and has don't fragment bit set.
3	Reassembly Problem - Destination could not reassemble due to missing fragments when time to live expired.
4	Gateway Congestion - Gateway discarded datagram due to congestion.

The error report is placed in a datagram with the following values in the internet header fields:

Version: Same as the datagram in error.
IHL: As computed.
Type of Service: Zero.
Total Length: As computed.
Identification: A new identification is selected.
Flags: Zero.
Fragment Offset: Zero.
Time to Live: Sixty.
Protocol: Same as the datagram in error.
Header Checksum: As computed.
Source Address: Address of the error reporting module.
Destination Address: Source address of the datagram in error.
Options: The General Error Report Option.
Padding: As needed.

Not copied on fragmentation, goes with first fragment.

Internet Timestamp

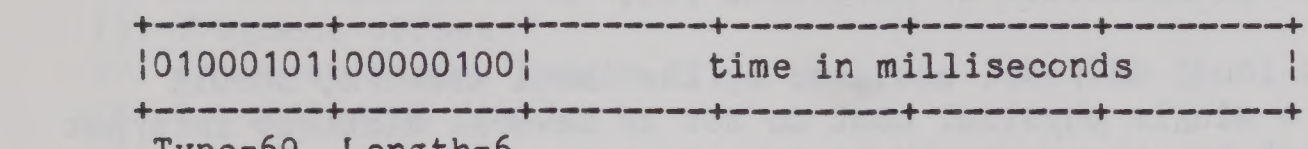


Type=68 Length=6

The data of the timestamp is a 32 bit time measured in milliseconds.

Not copied on fragmentation, goes with first fragment

Satellite Timestamp



The data of the timestamp is a 32 bit time measured in milliseconds.

Not copied on fragmentation, goes with first fragment

Padding: variable

The internet header padding is used to ensure that the internet header ends on a 32 bit boundary. The padding is zero.

3.2. Discussion

The implementation of a protocol must be robust. Each implementation must expect to interoperate with others created by different individuals. While the goal of this specification is to be explicit about the protocol there is the possibility of differing interpretations. In general, an implementation should be conservative in its sending behavior, and liberal in its receiving behavior. That is, it should be careful to send well-formed datagrams, but should accept any datagram that it can interpret (e.g., not object to technical errors where the meaning is still clear).

The basic internet service is datagram oriented and provides for the fragmentation of datagrams at gateways, with reassembly taking place at the destination internet protocol module in the destination host. Of course, fragmentation and reassembly of datagrams within a network or by private agreement between the gateways of a network is also allowed since this is transparent to the internet protocols and the higher-level protocols. This transparent type of fragmentation and reassembly is termed "network-dependent" (or intranet) fragmentation and is not discussed further here.

Internet addresses distinguish sources and destinations to the host level and provide a protocol field as well. It is assumed that each protocol will provide for whatever multiplexing is necessary within a host.

Addressing

The 8 bit network number, which is the first octet of the address, has a value as specified in reference [6].

The 24 bit local address, assigned by the local network, should allow for a single physical host to act as several distinct internet hosts. That is, there should be mapping between internet host addresses and network/host interfaces that allows several internet addresses to correspond to one interface. It should also be allowed for a host to have several physical interfaces and to treat the datagrams from several of them as if they were all addressed to a single host. Address mappings between internet addresses and addresses for ARPANET, SATNET, PRNET, and other networks are described in reference [4].

Fragmentation and Reassembly.

The internet identification field (ID) is used together with the source and destination address, and the protocol fields, to identify datagram fragments for reassembly.

The More Fragments flag bit (MF) is set if the datagram is not the last fragment. The Fragment Offset field identifies the fragment location, relative to the beginning of the original unfragmented datagram. Fragments are counted in units of 8 octets. The fragmentation strategy is designed so than an unfragmented datagram has all zero fragmentation information (MF = 0, fragment offset = 0). If an internet datagram is fragmented, its data portion must be broken on 8 octet boundaries.

This format allows 2**13 = 8192 fragments of 8 octets each for a total of 65,536 octets. Note that this is consistent with the the datagram total length field.

When fragmentation occurs, some options are copied, but others remain with the first fragment only.

Every internet module must be able to forward a datagram of 68 octets without further fragmentation. This is because an internet header may be up to 60 octets, and the minimum fragment is 8 octets.

Every internet destination must be able to receive a datagram of 576 octets either in one piece or in fragments to be reassembled.

The fields which may be affected by fragmentation include:

(1) options field
(2) more fragments flag
(3) fragment offset
(4) internet header length field
(5) total length field
(6) header checksum

If the Don't Fragment flag (DF) bit is set, then internet fragmentation of this datagram is NOT permitted, although it may be discarded. This can be used to prohibit fragmentation in cases where the receiving host does not have sufficient resources to reassemble internet fragments.

General notation in the following pseudo programs: "=<" means "less than or equal", "#" means "not equal", "=" means "equal", "<-" means "is set to". Also, "x to y" includes x and excludes y; for example, "4 to 7" would include 4, 5, and 6 (but not 7).

Fragmentation Procedure

The maximum sized datagram that can be transmitted through the next network is called the maximum transmission unit (MTU).

If the total length is less than or equal the maximum transmission unit then submit this datagram to the next step in datagram processing; otherwise cut the datagram into two fragments, the first fragment being the maximum size, and the second fragment being the rest of the datagram. The first fragment is submitted to the next step in datagram processing, while the second fragment is submitted to this procedure in case it still too large.

Notation:

FO - Fragment Offset
IHL - Internet Header Length
MF - More Fragments flag
TL - Total Length
OFO - Old Fragment Offset
OIHL - Old Internet Header Length
OMF - Old More Fragments flag
OTL - Old Total Length
NFB - Number of Fragment Blocks
MTU - Maximum Transmission Unit

Procedure:

```
IF TL =< MTU THEN Submit this datagram to the next step
     in datagram processing ELSE
To produce the first fragment:
(1)  Copy the original internet header;
(2)  OIHL <- IHL; OTL <- TL; OFO <- FO; OMF <- MF;
(3)  NFB <- (MTU-IHL*4)/8;
(4)  Attach the first NFB*8 data octets;
(5)  Correct the header:
     MF <- 1;  TL <- (IHL*4)+(NFB*8);
     Recompute Checksum;
(6)  Submit this fragment to the next step in
     datagram processing;
To produce the second fragment:
(7)  Selectively copy the internet header (some options
     are not copied, see option definitions);
(8)  Append the remaining data;
(9)  Correct the header:
     IHL <- (((OIHL*4)-(length of options not copied))+3)/4;
     TL <- OTL - NFB*8 - (OIHL-IHL)*4);
     FO <- OFO + NFB;  MF <- OMF;  Recompute Checksum;
(10) Submit this fragment to the fragmentation test; DONE.
```

Reassembly Procedure

For each datagram the buffer identifier is computed as the concatenation of the source, destination, protocol, and identification fields. If this is a whole datagram (that is both the fragment offset and the more fragments fields are zero), then any reassembly resources associated with this buffer identifier are released and the datagram is forwarded to the next step in datagram processing.

If no other fragment with this buffer identifier is on hand then reassembly resources are allocated. The reassembly resources consist of a data buffer, a header buffer, a fragment block bit table, a total data length field, and a timer. The data from the fragment is placed in the data buffer according to its fragment offset and length, and bits are set in the fragment block bit table corresponding to the fragment blocks received.

If this is the first fragment (that is the fragment offset is zero) this header is placed in the header buffer. If this is the last fragment (that is the more fragments field is zero) the total data length is computed. If this fragment completes the datagram (tested by checking the bits set in the fragment block table), then the datagram is sent to the next step in datagram

processing; otherwise the timer is set to the maximum of the current timer value and the value of the time to live field from this fragment; and the reassembly routine gives up control.

If the timer runs out, the all reassembly resources for this buffer identifier are released. The initial setting of the timer is a lower bound on the reassembly waiting time. This is because the waiting time will be increased if the Time to Live in the arriving fragment is greater than the current timer value but will not be decreased if it is less. The maximum this timer value could reach is the maximum time to live (approximately 4.25 minutes). The current recommendation for the initial timer setting is 15 seconds. This may be changed as experience with this protocol accumulates. Note that the choice of this parameter value is related to the buffer capacity available and the data rate of the transmission medium; that is, data rate times timer value equals buffer size (e.g., 10Kb/s X 15s = 150Kb).

Notation:

```
FO    -  Fragment Offset
IHL   -  Internet Header Length
MF    -  More Fragments flag
TTL   -  Time To Live
NFB   -  Number of Fragment Blocks
TL    -  Total Length
TDL   -  Total Data Length
BUFID -  Buffer Identifier
RCVBT -  Fragment Received Bit Table
TLB   -  Timer Lower Bound
```

```
Procedure:

(1)  BUFID <- source|destination|protocol|identification;
(2)  IF FO = 0 AND MF = 0
(3)     THEN IF buffer with BUFID is allocated
(4)             THEN flush all reassembly for this BUFID;
(5)          Submit datagram to next step; DONE.
(6)     ELSE IF no buffer with BUFID is allocated
(7)             THEN allocate reassembly resources
                     with BUFID;
                     TIMER <- TLB; TDL <- 0;
(8)          put data from fragment into data buffer with
             BUFID from octet FO*8 to
                                  octet (TL-(IHL*4))+FO*8;
(9)          set RCVBT bits from FO
                                 to FO+((TL-(IHL*4)+7)/8);
(10)         IF MF = 0 THEN TDL <- TL-(IHL*4)+(FO*8)
(11)         IF FO = 0 THEN put header in header buffer
(12)         IF TDL # 0
(13)          AND all RCVBT bits from 0
                                      to (TDL+7)/8 are set
(14)            THEN TL <- TDL+(IHL*4)
(15)                 Submit datagram to next step;
(16)                 free all reassembly resources
                     for this BUFID; DONE.
(17)         TIMER <- MAX(TIMER,TTL);
(18)         give up until next fragment or timer expires;
(19) timer expires: flush all reassembly with this BUFID; DONE.
```

In the case that two or more fragments contain the same data either identically or through a partial overlap, this procedure will use the more recently arrived copy in the data buffer and datagram delivered.

Identification

The choice of the Identifier for a datagram is based on the need to provide a way to uniquely identify the fragments of a particular datagram. The protocol module assembling fragments judges fragments to belong to the same datagram if they have the same source, destination, protocol, and Identifier. Thus, the sender must choose the Identifier to be unique for this source, destination pair and protocol for the time the datagram (or any fragment of it) could be alive in the internet.

It seems then that a sending protocol module needs to keep a table of Identifiers, one entry for each destination it has communicated with in the last maximum packet lifetime for the internet.

However, since the Identifier field allows 65,536 different values, some host may be able to simply use unique identifiers independent of destination.

It is appropriate for some higher level protocols to choose the identifier. For example, TCP protocol modules may retransmit an identical TCP segment, and the probability for correct reception would be enhanced if the retransmission carried the same identifier as the original transmission since fragments of either datagram could be used to construct a correct TCP segment.

Type of Service

The type of service (TOS) is for internet service quality selection. The type of service is specified along the abstract parameters precedence, reliability, and speed. A further concern is the possibility of efficient handling of streams of datagrams. These abstract parameters are to be mapped into the actual service parameters of the particular networks the datagram traverses.

Precedence. An independent measure of the importance of this datagram.

Stream or Datagram. Indicates if there will be other datagrams from this source to this destination at regular frequent intervals justifying the maintenance of stream processing information.

Reliability. A measure of the level of effort desired to ensure delivery of this datagram.

Speed over Reliability. Indicates the relative importance of speed and reliability when a conflict arises in meeting the pair of requests.

Speed. A measure of the importance of prompt delivery of this datagram.

For example, the ARPANET has a priority bit, and a choice between "standard" messages (type 0) and "uncontrolled" messages (type 3), (the choice between single packet and multipacket messages can also be considered a service parameter). The uncontrolled messages tend to be less reliably delivered and suffer less delay. Suppose an internet datagram is to be sent through the ARPANET. Let the internet type of service be given as:

Precedence:	5
Stream:	0
Reliability:	1
S/R:	1
Speed:	1

The mapping of these parameters to those available for the ARPANET would be to set the ARPANET priority bit on since the Internet priority is in the upper half of its range, to select uncontrolled messages since the speed and reliability requirements are equal and speed is preferred.

The following chart presents the recommended mappings from the internet protocol type of service into the service parameters actually available on the ARPANET, the PRNET, and the SATNET:

Application	INTERNET	ARPANET	PRNET	SATNET
TELNET on TCP	S/D:stream R:normal S/R:speed S:fast	T: 3 S: S	R: ptp A: no	T: block D: min H: inf R: no
FTP on TCP	S/D:stream R:normal S/R:rlblt S:normal	T: 0 S: M	R: ptp A: no	T: block D: normal H: inf R: no
interactive narrow band speech	S/D:strm* R:least P:speed S:asap	T: 3 S: S	R: ptp A: no	T: stream D: min H: short R: no
datagram	S/D:dtgrm R:normal S/R:speed S:fast	T: 3 or 0 S: S or M	R:station A: no	T: block D: min H: short R: no

key:	S/D=strm/dtgrm	T=type	R=route	T=type
	R=reliability	S=size	A=ack	D=delay
	S/R=speed/rlblt			H=holding time
	S=speed			R=reliability
	*=requires stream set up			

Time to Live

The time to live is set by the sender to the maximum time the datagram is allowed to be in the internet system. If the datagram is in the internet system longer than the time to live, then the datagram should be destroyed. This field should be decreased at each point that the internet header is processed to reflect the time spent processing the datagram. Even if no local information is available on the time actually spent, the field should be decremented by 1. The time is measured in units of seconds (i.e. the value 1 means one second). Thus, the maximum time to live is 255 seconds or 4.25 minutes.

Options

The options are just that, optional. That is, the presence or absence of an option is the choice of the sender, but each internet module must be able to parse every option. There can be several options present in the option field.

The options might not end on a 32-bit boundary. The internet header should be filled out with octets of zeros. The first of these would be interpreted as the end-of-options option, and the remainder as internet header padding.

Every internet module must be able to act on the following options: End of Option List (0), No Operation (1), Source Route (3), Return Route (7), General Error Report (33), and Internet Timestamp (68). The Security Option (2) is required only if classified or compartmented traffic is to be passed.

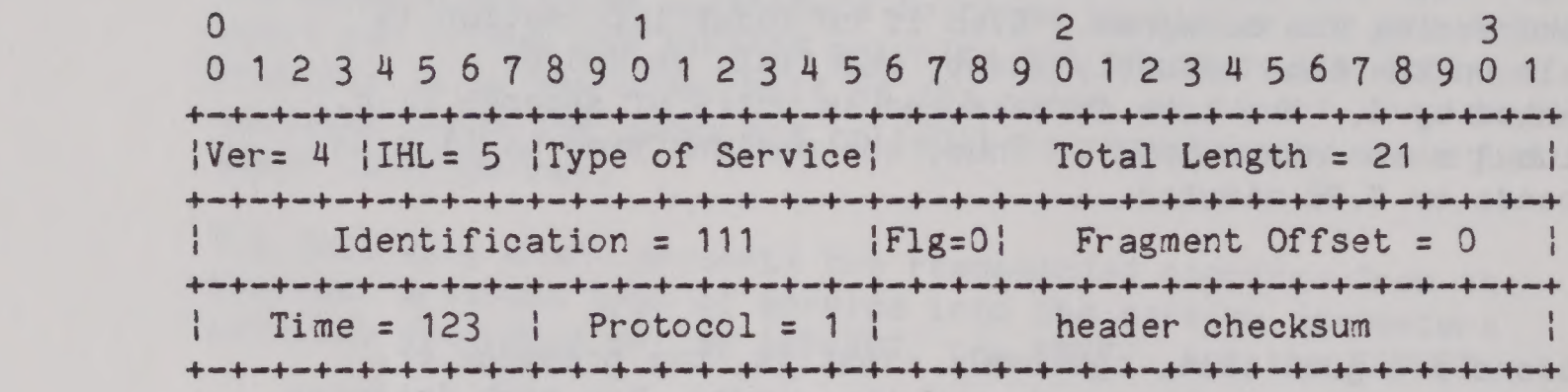
Checksum

The internet header checksum is recomputed if the internet header is changed. For example, a reduction of the time to live, additions or changes to internet options, or due to fragmentation. This checksum at the internet level is intended to protect the internet header fields from transmission errors.

3.3. Examples & Scenarios

Example 1:

This is an example of the minimal data carrying internet datagram:


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Ver= 4
IHL= 5
Type of Service
Total Length = 21
Identification = 111
Flg=0
Fragment Offset = 0
Time = 123
Protocol = 1
header checksum
source address
destination address
data

Example Internet Datagram

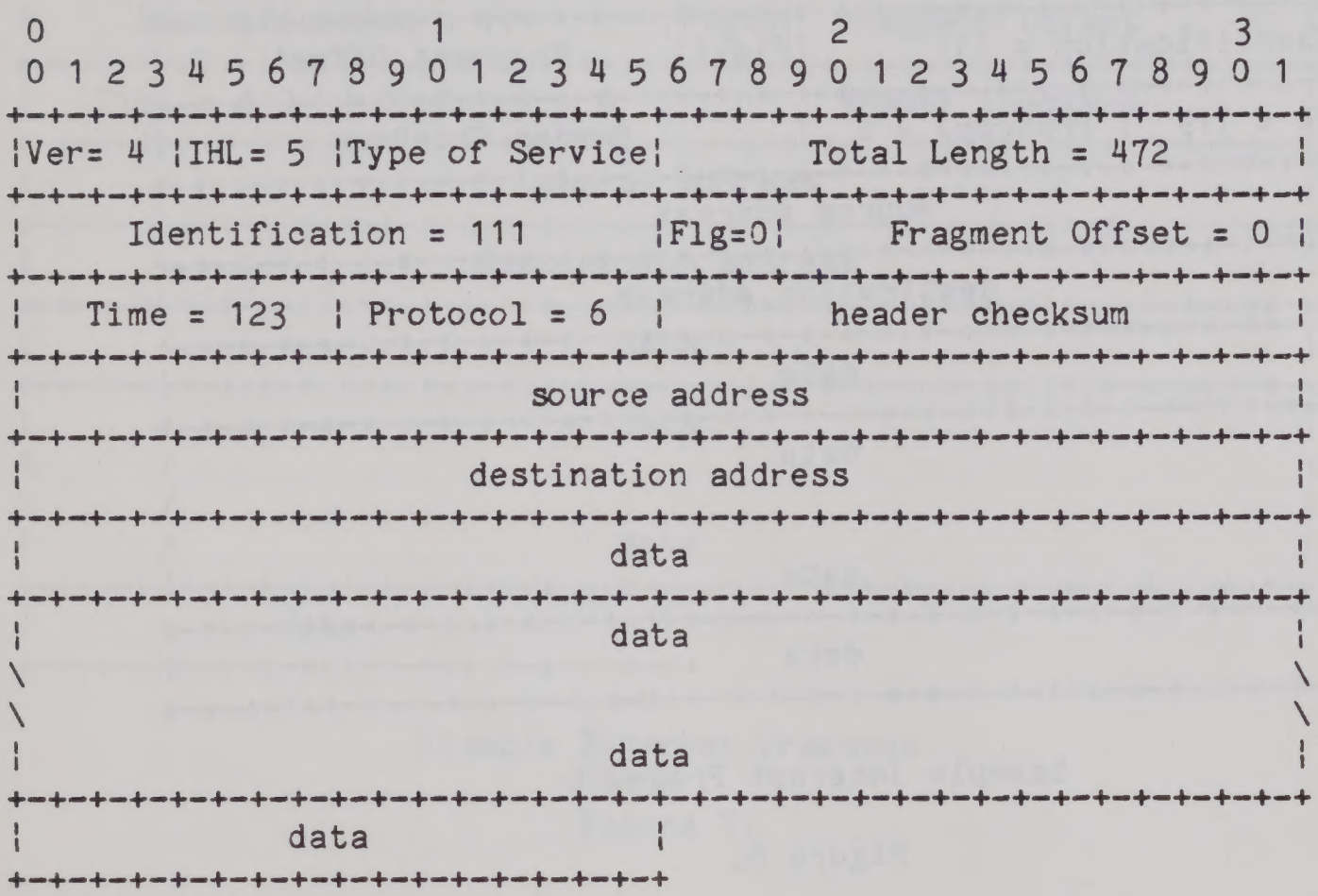
Figure 4.

Note that each tick mark represents one bit position.

This is a internet datagram in version 4 of internet protocol; the internet header consists of five 32 bit words, and the total length of the datagram is 21 octets. This datagram is a complete datagram (not a fragment).

Example 2:

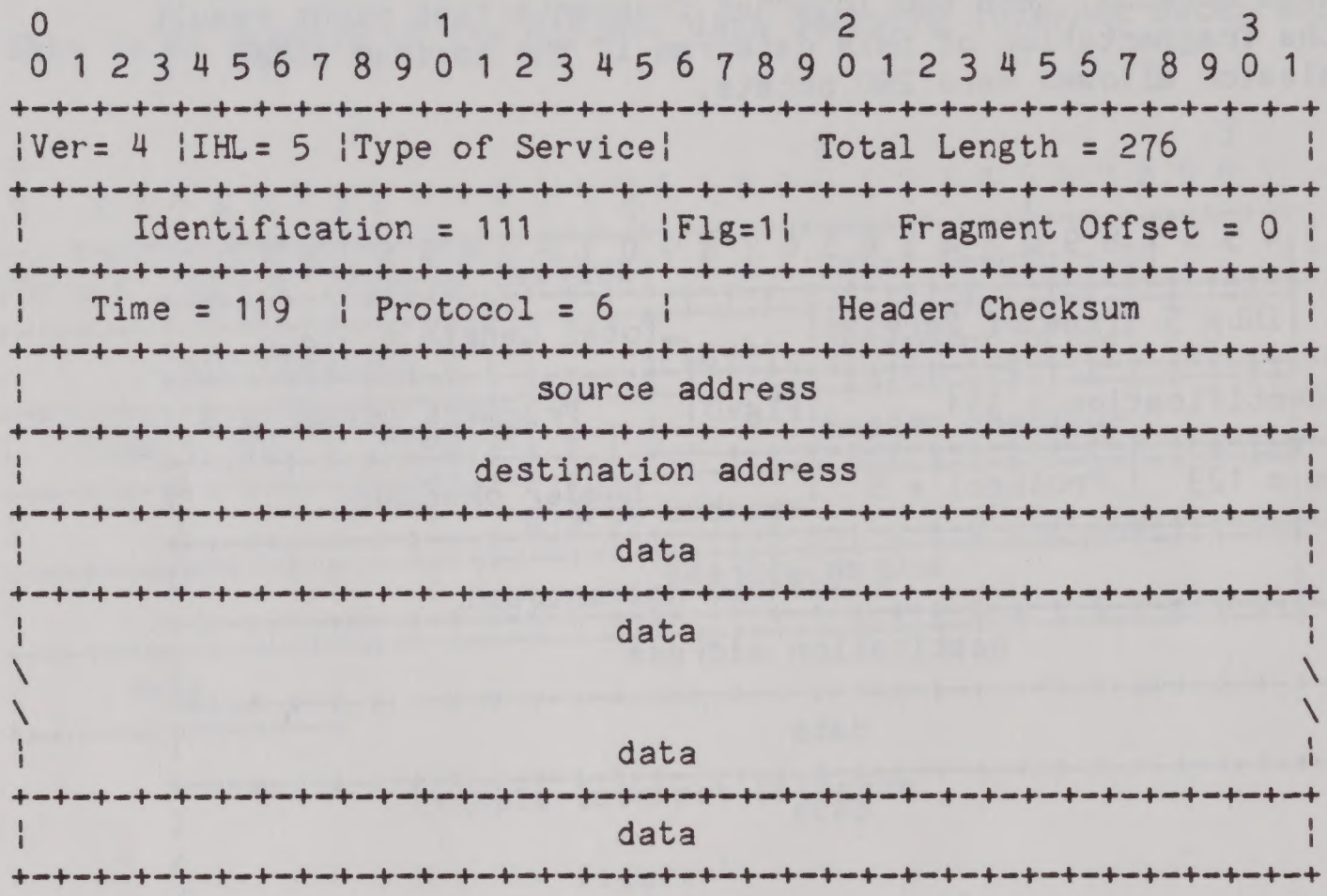
In this example, we show first a moderate size internet datagram (552 data octets), then two internet fragments that might result from the fragmentation of this datagram if the maximum sized transmission allowed were 280 octets.



Example Internet Datagram

Figure 5.

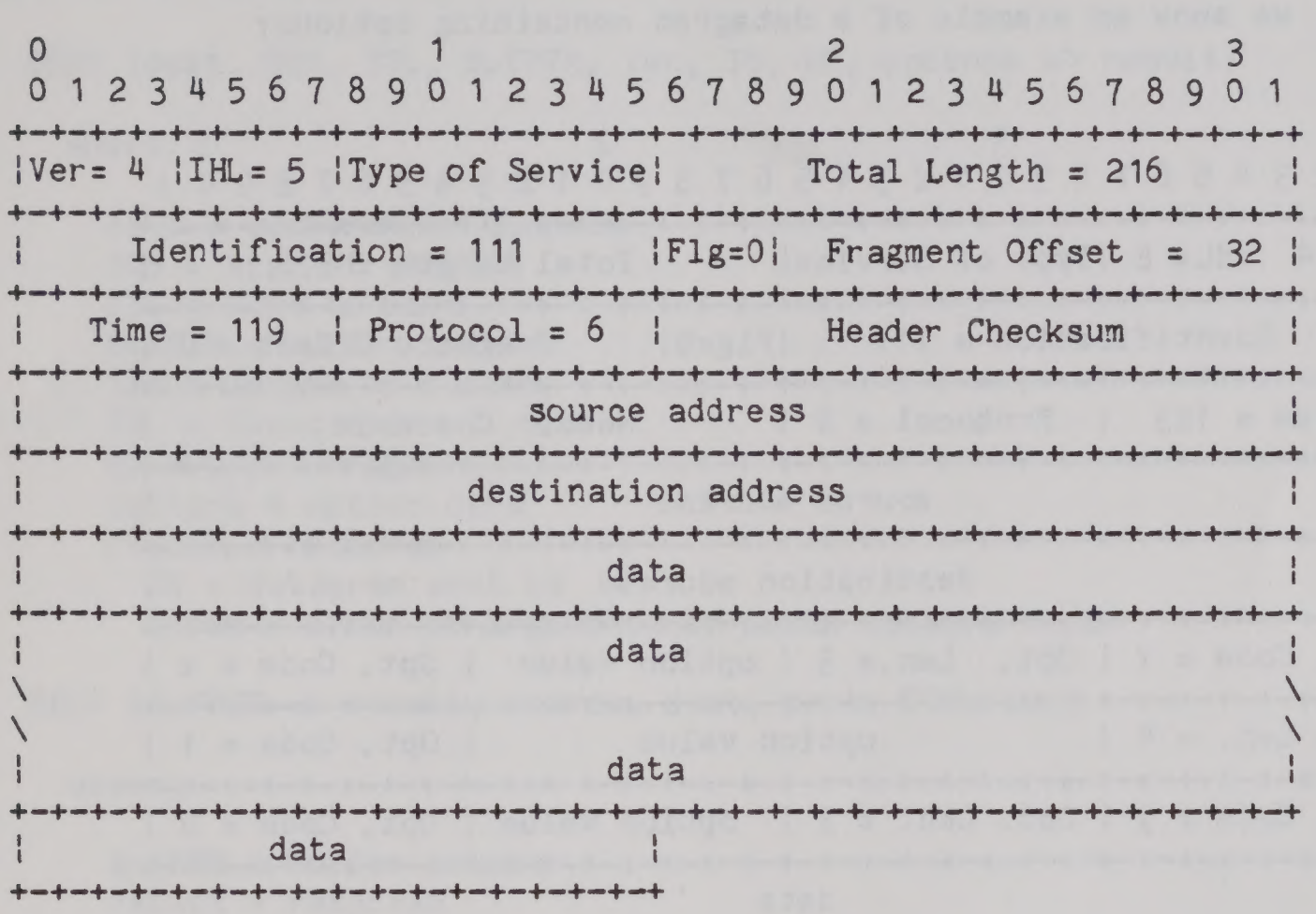
Now the first fragment that results from splitting the datagram after 256 data octets.



Example Internet Fragment

Figure 6.

And the second fragment.

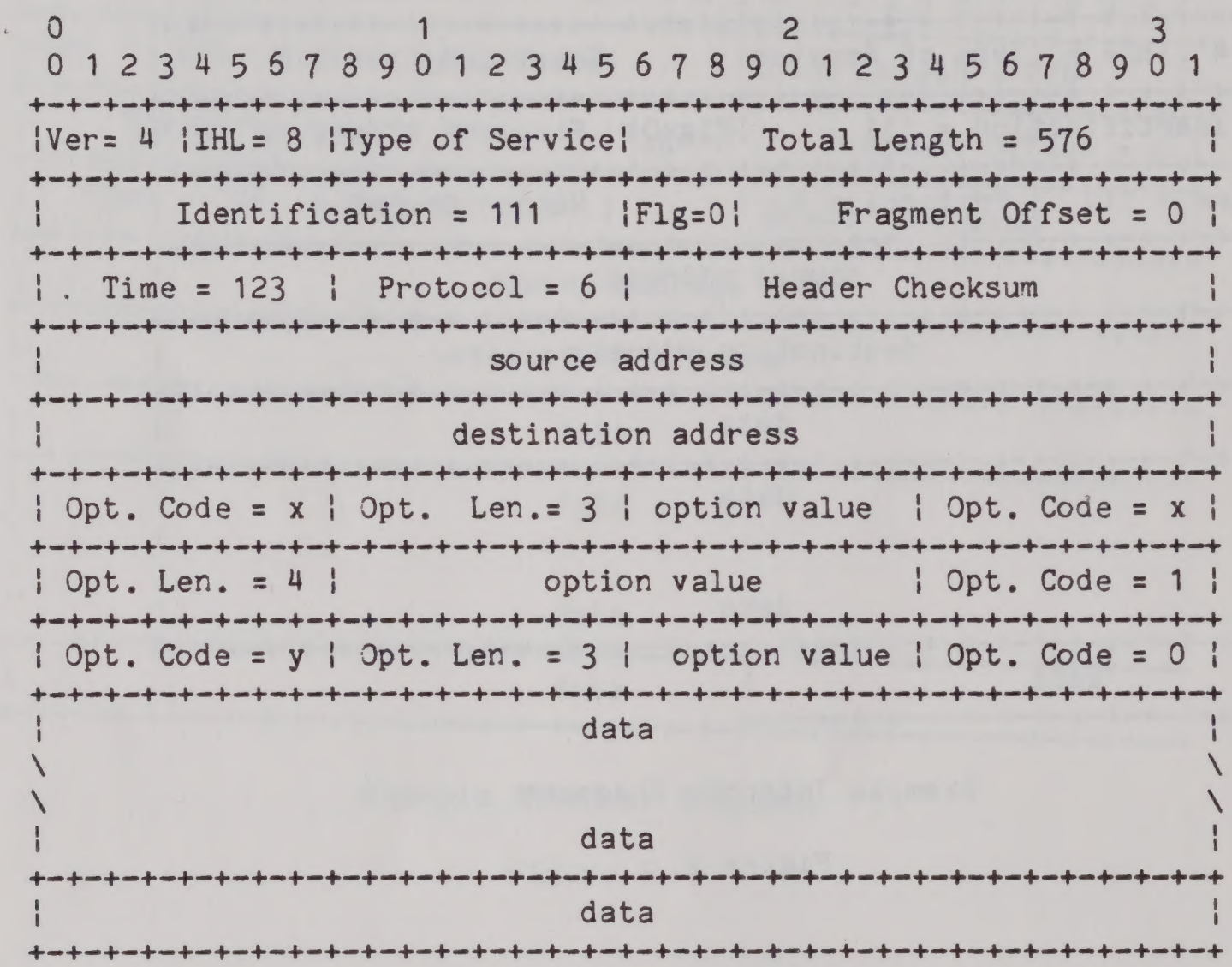

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Ver= 4
IHL= 5
Type of Service
Total Length = 216
Identification = 111
Flg=0
Fragment Offset = 32
Time = 119
Protocol = 6
Header Checksum
source address
destination address
data
data
data

Example Internet Fragment

Figure 7.

Example 3:

Here, we show an example of a datagram containing options:



Example Internet Datagram

Figure 8.

3.4. Interfaces

Internet protocol interfaces on one side to the local network and on the other side to either a higher level protocol or an application program. In the following, the higher level protocol or application program (or even a gateway program) will be called the "user" since it is using the internet module. Since internet protocol is a datagram protocol, there is minimal memory or state maintained between datagram transmissions, and each call on the internet protocol module by the user supplies all the necessary information.

For example, the following two calls satisfy the requirements for the user to internet protocol module communication ("=>" means returns):

```
SEND (dest, TOS, TTL, BufPTR, len, Id, DF, options => result)

  where:

    dest = destination address
    TOS = type of service
    TTL = time to live
    BufPTR = buffer pointer
    len = length of buffer
    Id  = Identifier
    DF = Don't Fragment
    options = option data
    result = response
      OK = datagram sent ok
      Error = error in arguments or local network error

RECV (BufPTR => result, source, dest, prot, TOS, len)

  where:

    BufPTR = buffer pointer
    result = response
      OK = datagram received ok
      Error = error in arguments
    source = source address
    dest = destination address
    prot = protocol
    TOS = type of service
    len = length of buffer
```

When the user sends a datagram, it executes the SEND call supplying all the arguments. The internet protocol module, on receiving this call, checks the arguments and prepares and sends the message. If the arguments are good and the datagram is accepted by the local network, the call returns successfully. If either the arguments are bad, or the datagram is not accepted by the local network, the call returns unsuccessfully. On unsuccessful returns, a reasonable report should be made as to the cause of the problem, but the details of such reports are up to individual implementations.

When a datagram arrives at the internet protocol module from the local network, either there is a pending RECV call from the user addressed or there is not. In the first case, the pending call is satisfied by passing the information from the datagram to the user. In the second case, the user addressed is notified of a pending datagram. If the

user addressed does not exist, an error datagram is returned to the sender, and the data is discarded.

The notification of a user may be via a pseudo interrupt or similar mechanism, as appropriate in the particular operating system environment of the implementation.

A user's RECV call may then either be immediately satisfied by a pending datagram, or the call may be pending until a datagram arrives.

An implementation may also allow or require a call to the internet module to indicate interest in or reserve exclusive use of a class of datagrams (e.g., all those with a certain value in the protocol field).

GLOSSARY

1822
: BBN Report 1822, "The Specification of the Interconnection of a Host and an IMP". The specification of interface between a host and the ARPANET.

ARPANET message
: The unit of transmission between a host and an IMP in the ARPANET. The maximum size is about 1012 octets (8096 bits).

ARPANET packet
: A unit of transmission used internally in the ARPANET between IMPs. The maximum size is about 126 octets (1008 bits).

Destination
: The destination address, an internet header field.

DF
: The Don't Fragment bit carried in the flags field.

Flags
: An internet header field carrying various control flags.

Fragment Offset
: This internet header field indicates where in the internet datagram a fragment belongs.

header
: Control information at the beginning of a message, segment, datagram, packet or block of data.

Identification
: An internet header field carrying the identifying value assigned by the sender to aid in assembling the fragments of a datagram.

IHL
: The internet header field Internet Header Length is the length of the internet header measured in 32 bit words.

IMP
: The Interface Message Processor, the packet switch of the ARPANET.

Internet Address
: A four octet (32 bit) source or destination address consisting of a Network field and a Local Address field.

internet fragment
: A portion of the data of an internet datagram with an internet header.

internet datagram
: The unit of data exchanged between a pair of internet modules (includes the internet header).

ARPANET leader
: The control information on an ARPANET message at the host-IMP interface.

Local Address
: The address of a host within a network. The actual mapping of an internet local address on to the host addresses in a network is quite general, allowing for many to one mappings.

MF
: The More-Fragments Flag carried in the internet header flags field.

module
: An implementation, usually in software, of a protocol or other procedure.

more-fragments flag
: A flag indicating whether or not this internet datagram contains the end of an internet datagram, carried in the internet header Flags field.

NFB
: The Number of Fragment Blocks in a the data portion of an internet fragment. That is, the length of a portion of data measured in 8 octet units.

octet
: An eight bit byte.

Options
: The internet header Options field may contain several options, and each option may be several octets in length. The options are used primarily in testing situations, for example to carry timestamps.

Padding
: The internet header Padding field is used to ensure that the data begins on 32 bit word boundary. The padding is zero.

Protocol
: In this document, the next higher level protocol identifier, an internet header field.

Rest
: The 3 octet (24 bit) local address portion of an Internet Address.

RTP
: Real Time Protocol: A host-to-host protocol for communication of time critical information.

Source
: The source address, an internet header field.

TCP
: Transmission Control Protocol: A host-to-host protocol for reliable communication in internet environments.

TCP Segment
: The unit of data exchanged between TCP modules (including the TCP header).

Total Length
: The internet header field Total Length is the length of the datagram in octets including internet header and data.

Type of Service
: An internet header field which indicates the type (or quality) of service for this internet datagram.

User
: The user of the internet protocol. This may be a higher level protocol module, an application program, or a gateway program.

Version
: The Version field indicates the format of the internet header.

REFERENCES

[1] Cerf, V., "The Catenet Model for Internetworking," Information Processing Techniques Office, Defense Advanced Research Projects Agency, IEN 48, July 1978.

[2] Bolt Beranek and Newman, "Specification for the Interconnection of a Host and an IMP," BBN Technical Report 1822, May 1978 (Revised).

[3] Shoch, J., "Inter-Network Naming, Addressing, and Routing," COMPCON, IEEE Computer Society, Fall 1978.

[4] Postel, J., "Address Mappings," IEN 115, USC/Information Sciences Institute, August 1979.

[5] Shoch, J., "Packet Fragmentation in Inter-Network Protocols," Computer Networks, v. 3, n. 1, February 1979.

[6] Postel, J., "Assigned Numbers," RFC 762, IEN 127, USC/Information Sciences Institute, January 1980.